CS112 - Fall 2022

Lab06

Instructor: Paul Haskell

INTRODUCTION

In this lab, you will develop some classes for playing cards and for a deck of cards, with overloaded constructors. These classes should be useful to you in our upcoming first big project.

Classes for you to design

First, please create a `class Card` that represents a single card from a standard 52-card deck of playing cards.

Each card has a value (2,3,4,5,6,7,8,9,10,Jack,Queen,King,Ace) and a suit (Spades,Hearts,Clubs,Diamonds).

Your class should have several constructors:

- One that takes an integer (2 through 14) to set the card value (2 through Ace) and a String to set the suit. To set the suit, look at the first letter of the argument, ignoring upper/lowercase, and if it is {s,h,c,d} then set the suit accordingly
- Another constructor takes a string and parses it to set the card value and suit. If the first character in the string is 2 through 9 or J, Q, K, or A (ignore upper/lowercase), then set the card value in the obvious way. If the first character is '1', then the second character must be '0' to indicate a card value of 10. The following character should be {s,h,c,d} to specify the suit. Any additional characters can be ignored
- A third constructor takes two Strings: the first specifies the card value and the second specifies the card suit. The two strings are defined the same as for the second constructor
- If there is any error in the input to the constructor, the card value should be set to 0 and the suit should be set to "ERROR"
- Finally, create a default constructor (no arguments) that acts as if the inputs were in error

In addition to the constructors, please add the following methods:

- Value(): returns the card's value as an integer in the range from 2 through 14. Jack has value 11, Queen has value 12, King has value 13, and Ace has value 14.
- Suit(): returns the card's suit as a String. The suit should be represented as a single capital letter: "S", "H", "C", or "D"
- A method called print() that returns a String showing the value and then the suit, with no space between them. However, instead of printing 11 for Jack, 12 for Queen, etc instead print "J" for Jack, "Q" for Queen, "K" for King, and "A" for Ace. For example, this method should print "10D", "QC", etc.

Class Deck

Your class Deck will store 52 cards: 13 cards for each suit, with values from 2 through 14 (or 2 through Ace, if you prefer). To store these, you will use an array. We talked about arrays when we talked about the String command line arguments in `static public void main(String[] args)`

Here, you will want code that looks something like:

```
Card allMyCards[] = new Card[52];
```

This creates an array called `allMyCards` that has storage for 52 Cards. The storage slots in the array are not initialized yet, but you can access them by index, e.g.

```
allMyCards[0], allMyCards[1], … allMyCards[51]
```

To initialize each array entry, you use the new operator, for example:

```
allMyCards[0] = new Card(2, "spades");
```

Ok, now you have instructions on how to create an empty array and on how to initialize each entry in the array. But you probably don't want to copy and paste 52 different lines of code to initialize each entry. (For an array with 52 entries, the indices are 0 through 51.) Instead, use a `while()` loop, or maybe a few of them: one to set the suit and one to set the card value. Or you could have separate loops for each suit.

In the future, you will extend class Deck to do more, but for now just add a `print()` method. This method returns a `String` that contains each card value in the deck, with a space (" ") between the values.

Details of the Assignment

Please put your `class Card` and `class Deck` into a file called **MyCardDeck.java**. Of course, you also need a `class MyCardDeck` that will hold the `main()` function. In `main()` do the following:

- If there is exactly one command-line argument to the program, initialize a `Card` with the argument passed to the one-String constructor. Then print the `Card` value
  ```
  System.out.println(yourCard.print());
  ```
  and exit.
- If there are two or more command-line argument to the program, initialize a `Card` with the first two arguments, passed to the two-String constructor. Then print the `Card` value and exit.

- If there are zero arguments to the program, create an object of type `Deck`, print out the entire deck, then exit.

Reminder

Print exactly what was requested. Put useful comments into your code. Design and organize your code so that it is easy for others to understand.

Put your program **MyCardDeck.java** into a subdirectory called **Lab06** inside your MyWork directory, and remember to push your **Lab06** to GitHub before the deadline. This assignment must be turned in before Monday Sept 18 at 11:59pm.

Conclusion

The `MyCardDeck.java` program and classes gave you some hands-on experience writing overloaded constructors and also writing to handle errors in user inputs.

Grading Rubric

The `MyCardDeck` program is worth 40 points:

- Two points if it compiles
- 14 points if the entire card deck prints correctly
- Two points each for 7 separate test cases that create and print a single card value. Many of these cases will be error handling cases.
- Zero to 10 points based on the grader's evaluation of MyCardDeck's software clarity and design quality